New SAM Schema at DØ: Description & Requirements

Diana Bonham,
Lauri Loebel Carpenter,
Anil Kumar, Adam Lyon[*]*, Carmenita Moore,*
Wyatt Merritt, Jeremy Simmons[†]*,*
Julie Trumbo, Stephen White, Sinisa Veseli

Fermilab

June 26, 2003

1 Introduction

The new SAM Schema aims to solve several problems:

- Use of place holder data due to attribute restrictions that are not appropriate for all file types.

- Inability to assign more than one run number to a file (especially important for files that are merged across many runs).

---

[*] For comments/corrections/questions regarding this document, send Adam e-mail at lyon@fnal.gov
[†] Consultant from Piocon Technologies, Naperville, IL

- Difficulty in grouping data files into data sets for easy access. Right now, users must remember valid reconstruction versions in order to define their datasets.

- Important file information is spread throughout the database. For example, to determine if a file is Monte Carlo or Collider Data requires a query with several joins.

- Incorporate CDF luminosity information for their files.

The original design idea is described in DØ Note 4083 (http://www-d0.fnal.gov/cgi-bin/d0note?4083) with separate DATA_FILES like tables for different types of files. That was seen as too large of a change, and so a compromise was reached. The design now is to have each file in DATA_FILES carry an attribute that describes the file type. Database triggers to validate specific DATA_FILE attributes will be discussed and coordinated jointly by the applications and database groups when dbserver code modifications are being developed.

This document describes this schema update as well as changes that will be necessary to applications, dbservers, etc.

The entire SAM DB schema ER diagram is too large to put in this document. See the latest version on the web at http://d0db.fnal.gov/sam/doc/design/sam_entities/er_diagram.ps.

2   File Meta-data

The DATA_FILES table holds most of the file information.

2.1 File Types
We identify five types of data files that are to be stored in the DATA_FILES table. The abbreviations are for reference in this document and are not meant to be in the database.

- *nonPhysicsGeneric (NPG)*: Generic non-event files (perhaps log files).

- *colliderImport (CI):* Event data from collisions that were brought into SAM without a SAM project (*e.g.* from the online system). At DØ these would be raw files.

- *simulationImport (SI):* Event data from Monte Carlo that were not produced by a SAM project.

- *derivedCollider (DC):* Event data produced by running a SAM project over collider data.

- *derivedSimulation (DS):* Event data produced by running a SAM project over Monte Carlo data.

- *physicsGeneric (PG):* Event data produced for personal purposes (storage of personal skims or Monte Carlo). These files may not have all meta-data.

- *cdfDataSet:* Used only by CDF.

- *cdfFileSet:* Used only by CDF.

The possible file types are stored in the new `FILE_TYPE` table. The file type ID is then stored in the `DATA_FILES` table in the `FILE_TYPE_ID` attribute. The file types will be stored with mixed case, though the DB server will transform them to lower case for querying purposes.

2.2 Attributes for `DATA_FILES`

The required and optional attributes in `DATA_FILES` now depend on the file type. Only attributes that are required for all file types are non-nullable. The following table describes all of the attributes for `DATA_FILES`. Note the abbreviations for the file types above. In the table, *R* indicates the attribute is required, *O* indicates optional, and *N/A* indicates the attribute is not applicable to that particular file type.

Attribute	*NPG*	*CI*	*SI*	*DC*	*DS*	*PG*	*Notes*
FILE_ID	R	R	R	R	R	R	**Non-nullable attribute.**
FILE_TYPE_ID	R	R	R	R	R	R	**Non-nullable attribute.** New attribute.
FILE_NAME	R	R	R	R	R	R	**Non-nullable attribute.**
FORMAT_INFO	R	R	R	R	R	R	**Non-nullable attribute.** Was nullable.
KBYTE_FILE_SIZE	R	R	R	R	R	R	**Non-nullable attribute.**
FILE_SIZE	R	R	R	R	R	R	**Non-nullable attribute.** Was nullable.
FILE_SIZE_UNITS_ID	R	R	R	R	R	R	**Non-nullable attribute.** Replaces

Attribute	NPG	CI	SI	DC	DS	PG	Notes
							FILE_UNITS
CRC_TYPE	R	R	R	R	R	R	**Non-nullable attribute.** Was nullable.
CRC_VALUE	R	R	R	R	R	R	**Non-nullable attribute.** Was nullable.
CREATE_USER	R	R	R	R	R	R	**Non-nullable attribute.**
CREATE_DATE	R	R	R	R	R	R	**Non-nullable attribute.**
UPDATE_USER	R	R	R	R	R	R	**Non-nullable attribute.**
UPDATE_DATE	R	R	R	R	R	R	**Non-nullable attribute.**
FILE_CONTENTS_ STATUS_ID	R	R	R	R	R	R	**Non-nullable attribute.** Was nullable.
DATA_TIER	N/A	R	R	R	R	R	All event data must have a data tier. This attribute does not apply to non-event data. Was non-nullable.
APPL_FAMILY_ID	N/A	R	R	R	R	O	Was non-nullable.
FILE_PARTITION	N/A	R	N/A	N/A	N/A	N/A	At DØ, only raw data files have a partition.
PROCESS_ID	N/A	N/A	N/A	R	R	O	Non-imported event data have process IDs. Was non-nullable.
RESPONSIBLE_WORKING_ GROUP_ID	O	N/A	R	R	R	O	New attribute. Indicates the group (e.g. W/Z) responsible for

Attribute	NPG	CI	SI	DC	DS	PG	Notes
							the data (e.g. W/Z) as opposed to the group who produced the data (e.g. MC)
STREAM_ID	N/A	R	O	R	O	O	Monte Carlo files may or may not be streamed. Was non-nullable.
EVENT_COUNT	N/A	R	R	R	R	O	Was non-nullable.
FIRST_EVENT_NUMBER	N/A	R	R	R	R	O	Difficult to fill for old files. Was non-nullable.
LAST_EVENT_NUMBER	N/A	R	R	R	R	O	Difficult to fill for old files. Was non-nullable.
START_TIME	N/A	R	N/A	N/A	N/A	N/A	At DØ, only raw files have a valid start time. Was non-nullable.
END_TIME	N/A	R	N/A	N/A	N/A	N/A	At DØ, only raw files have a valid end time. Was non-nullable.
LUM_SUM_ONLINE	N/A	?	?	?	?	?	New attribute. DØ does not store luminosity information at the file level.
LUM_SUM_OFFLINE	N/A	?	?	?	?	?	New attribute. DØ does not store luminosity information at the file level.
LOW_RUN	N/A	R	R	R	R	O	New attribute. Since this

Attribute	NPG	CI	SI	DC	DS	PG	Notes
							information is needed to fill the new RUNS mapping table anyway, it could be filled into this attribute
HIGH_RUN	N/A	R	R	R	R	O	New attribute. "ditto"

2.2.1 Removed attributes

The following attributes are removed from the DATA_FILES table,

- FILE_STATUS. This attribute is deprecated from a previous schema cut. Any data may be discarded during the schema migration.

- FILE_AVAILABILITY_STATUS. This attribute is deprecated. File availability information is saved with location and station information. Any data may be discarded during the schema migration.

- RUN_ID. This attribute is replaced by the DATAFILE_RUNS table. Data should be saved as discussed in section 4.2.

- LUM_MIN, LUM_MAX. The data from these attributes are in the new DATAFILE_LUMBLOCK table (see section 5). Note that for the schema migration, any data in these attributes should be copied to the new table.

- MINBIAS_NUMBER, MINBIAS_TYPE, PHYSICS_PROCESS_ID. These attributes were meant to store Monte Carlo specific information. Such information is duplicated in the MC parameters.

2.3 Implementation

Implementation of this new schema will require changes to the database, database servers and applications.

The DB server methods and applications that deal with saving files in SAM will have to pay attention to the FILE_TYPE and require other attributes as appropriate.

In order to keep the DB servers as flexible as possible, the attribute constraints will be coded in a configuration file loaded at runtime. The DB server could thus be easily tailored for DØ and CDF.

2.3.1 New FILE_TYPE_ID attribute

The `FILE_TYPE_ID` attribute is new to the schema. DB servers will need to fill this attribute and query applications will need to use it. This change permeates most applications (MISWEB, Dataset Definition Editor, etc.) that query for files. The ID will point into a new `FILE_TYPE` table that specifies the valid file types as shown in section 2.1.

For the schema migration, the current files in the DB must be assigned a file type. This assignment may be determined from the file data-tier and the run type. To get the run type, determine the `RUN_ID` of the file and look it up in the `RUNS` table. The `RUN_TYPE_ID` has the run type information.

CDF and DØ will migrate to the file type differently. The rules for DØ are as follows:

- Data-tiers of *generated* and *generated-bygroup* have **simulationImport** file type.

- Data-tiers of *simulated*, *simulated-bygroup*, *digitized* and *digitized-bygroup* have **derivedSimulation** file type.

- Data-tiers of *raw* and *raw-bygroup* have **colliderImport** file type.

- Data-tier of *triggersimulated* has **derivedSimulation** file type if the file's run type is Monte Carlo. Otherwise, the file type is **derivedCollider**. The latter case would be a collider data file processed with a simulated trigger list. Since the event data itself is collider, the resulting file is also deemed collider.

- Data-tiers of *reconstructed, reconstructed-bygroup, thumbnail, thumbnail-bygroup, filtered-raw, filtered-reco, filtered-root, filtered-thumbnail, root-bygroup, root-tuple, root-typle-bygroup, virtual-filtered-reco, virtual-filtered-root, virtual-root, virtual-thumbnail, v-filtered-thumbnail* have **derivedSimulation** file type if the run type is Monte Carlo. Otherwise, the file type is **derivedCollider**.

- Data-tier of *unofficial-reco* has **physicsGeneric** file type.

- Data-tiers of *epics, sam-dbserver-log, sam-master-log, significant-event*, and *special* have **nonPhysicsGeneric** file type.

The rules for CDF are as follows:

- File status of *virtual:* with file name of six characters have the **cdfDataSet** file type; with file name of eight characters have the **cdfFileSet** type; otherwise file type is **nonPhysicsGeneric**.

- File status of *being imported* or *deleted:* **nonPhysicsGeneric** file type.

- File status of *available* with data-tier of *raw:* **colliderImport** file type if the file name has 17 characters; otherwise file type is **nonPhysicsGeneric**.

- File status of *available* with data-tier of *reconstructed:* **derivedCollider** file type if the file name has 17 characters; otherwise file type is **nonPhysicsGeneric**.

- File status of *available* with data-tiers of *generated or simulated:* **simluatedImport** file type if the file name has 17 characters; otherwise file type is **nonPhysicsGeneric**.

- File status of *available* with data-tier of *unidentified:* **nonPhysicsGeneric** file type (regardless of file name length)

2.3.2 FORMAT_INFO attribute

This attribute is meant to describe what application or tool is needed to read the file (for example, dspack, tar, root, gzip). At this time, the FORMAT_INFO attribute is not being filled. For the schema migration, it's easy to automatically deduce the file format. The rules for DØ are as follows:

- *digitized, digitized-bygroup, filtered-raw, filtered-reco, filtered-thumbnail, generated, generated-bygroup, raw, raw-bygroup, reconstructed, reconstructed-bygroup, simulated, simulated-bygroup, thumbnail, thumbnail-bygroup, triggersimulated, unofficial_reco* data tiers are all in the **DSPACK** format.

- *root-bygroup, root-tuple, root-tuple-bygroup, filtered-root* data tiers are all in the **ROOT** format.

- *v-filtered-thumbnail, virtual-filtered-reco, virtual-filtered-root, virtual-thumbnail* are all in the **ETHEREAL** format. (Entries of these data-tiers may be removed in a later schema cut).

- For all other data tiers (e.g. *epics, sam-dbserver-log, sam-master-log, significant-event, special*), the type depends on the file name. If the file name ends in "tar", then the format should be **TAR**. If the file name ends in "tar.gz" then the format should be **GZIPPED-TAR**. If the file ends in ".sta", then the format should be **RUN-1-STA**.

The DB server should add the correct FORMAT_INFO when it is easy to determine. Otherwise, it will have to use and require input from the user. The applications that store files into SAM will have to allow for such input for the appropriate non-event data tiers.

2.3.3 New RESPONSIBLE_WORKING_GROUP_ID attribute

This new attribute points into the WORKING_GROUP table and is meant to identify the group responsible for the contents of a particular data file in SAM. It is not meant for resource tracking, but rather tells the user who they can talk to if they have a question about a file. This attribute is different than the WORKING_GROUP_ID that may be obtained from the process information (that is the group that produced the file).

The DB server methods and applications that deal with storing files into SAM will have to deal with this attribute.

2.3.4 FILE_SIZE and like attributes

FILE_SIZE_UNITS_ID replaces the FILE_SIZE_UNITS attribute and points into a new FILE_SIZE_UNITS table.

The FILE_SIZE, FILE_SIZE_UNITS_ID and KBYTE_FILE_SIZE attributes are now all non-nullable. For the schema migration, if FILE_SIZE is null, replace it with the value from KBYTE_FILE_SIZE and set the FILE_SIZE_UNITS_ID to correspond to *Kbytes*.

2.3.5 CRC_TYPE and CRC_VALUE attributes

These fields have become non-nullable. For the schema migration, if CRC_VALUE is null, replace it with "unknown value". If CRC_TYPE is null, replace it with "unknown crc type".

2.3.6 FIRST_EVENT_NUMBER and LAST_EVENT_NUMBER attributes

These attributes are now nullable (were non-nullable). This information should be determined as meta-data by applications writing files to be stored in SAM. DB server methods and SAM applications must accept this information for storage in the database. However, it will be difficult to fill in these attributes for files already stored in SAM without actually reading each file and determining the event numbers. Perhaps this is more trouble that it's worth. Given that the farm does not process events in order, perhaps these fields are not so useful for DØ and should be left null always.

2.3.7 New LUM_SUM_ONLINE and LUM_SUM_OFFLINE attributes

DØ does not calculate luminosity on a per-file basis. These attributes should be null always or filled with zeros.

2.3.8 New LOW_RUN and HIGH_RUN attributes

These attributes represent the lowest and highest run numbers for a given file. While these are redundant with the information in the `DATA_FILES_RUNS` mapping table (see below), they may make some queries simpler. Furthermore, since this information is needed anyway to fill the runs mapping table, it's not a huge deal. Of course DB server methods and application programs will need to determine the high/low run numbers and fill the attributes.

3   Valid Data Groups

Currently, users must remember all of the versions of the reconstruction program that correspond to good data. When we re-reconstruct a set of data, users must know that a certain reco version is now bad and should not be used. Instead of having users remember all this information, SAM can keep track of data that is valid for different purposes through Valid Data Groups. Valid Data Groups are a new feature introduced by this schema.

3.1   Structure

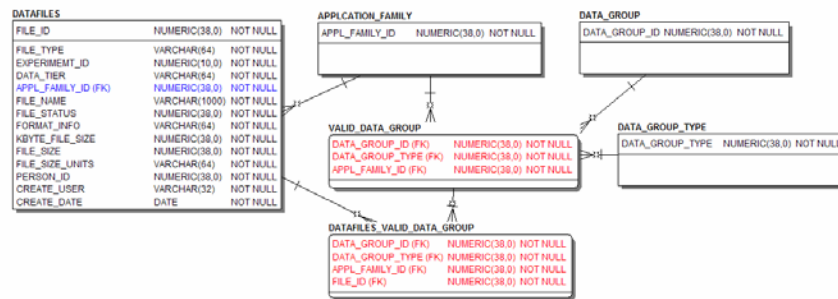The ER diagram for Valid Data Groups is shown below.

Figure 1: Valid data groups structure

Data files belonging to a valid data group have an entry in `DATA_FILES_VALID_DATA_GROUP`, the mapping table between `DATA_FILES` and `VALID_DATA_GROUP`. A valid data group may have a name and a type defined in their respective tables. Furthermore, a valid data group is associated with one or more application families.

An example valid data group is "p13Moriond2003", containing all files valid for analysis for Moriond 2003. A user would only have to remember this valid data group name to gain access to these data.

3.2   Implementation

Several additions to the SAM DB server and application software are needed to make valid data groups work:

- There must be automated mechanisms for managing the files that belong to different valid data groups. For example, files rolling off the farm should automatically be added to the "current" valid data group.

- For reprocessing operations, there must be mechanisms for removing the original files from the group and adding the new files.

- There must be easy query mechanisms for viewing what files belong to a valid data group.

- Valid data group queries must be added to the current data set definition mechanisms.

4   Runs

In the current SAM database, a file can be associated with only one run number. At DØ, this restriction is fine for raw files and files produced by the reconstruction farm. But user skim files that may be placed back in SAM may span more than one run. To allow for multirun files, a mapping table between `DATA_FILES` and `RUNS` is added.
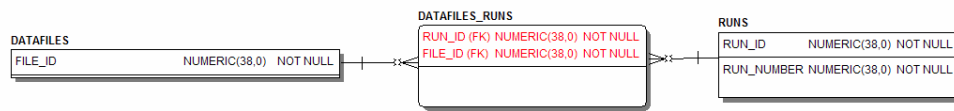
Figure 2: Connection to run numbers

4.1   Implementation

DB server methods and application programs that store data into SAM will now have to fill the mapping table. While doing so, `LOW_RUN` and `HIGH_RUN` from `DATA_FILES` should also be filled. Furthermore, query applications will now have to do the table join with the mapping table to determine the runs associated with a file.

4.2   Schema Migration

For event data, only thumbnail and thumbnail-bygroup data-tiers have files that actually correspond to data from more than one run (this is true for DØ, probably not for CDF). For all other data_tiers, just copy the value of the old `RUN_ID` attribute from `DATA_FILES` into the new `DATA_FILES_RUNS` table.

For a thumbnail or thumbnail-bygroup file, get all of the RAW file parents and put their run numbers into the DATA_FILES_RUNS table. Note that this may involve several hops through the lineage table.

1. Given a thumbnail file ID

2. Look up `FILE_ID_SOURCE` in `FILE_LINEAGES` where `FILE_ID_DEST` = the thumbnail file ID (this looks up the parents of the thumbnail file).

3. For each `FILE_ID_SOURCE`, look up the data-tier in `DATA_FILES`. If the data-tier is *raw*, then put its `RUN_ID` into the `DATA_FILES_RUNS` table and associate it with the thumbnail file ID.

4. If the data-tier is not raw, then look up `FILE_ID_SOURCE` in `FILE_LINEAGES` where `FILE_ID_DEST` = this fileID. (This looks up the parents of the non-raw file – another hop). Then repeat 3. above.

5   Luminosity Blocks

In the current SAM database, each file in `DATA_FILES` has `LUMBLOCK_MIN` and `LUMBLOCK_MAX` attributes. At DØ these attributes are only appropriate for raw files. CDF wants to store luminosity information in the SAM database itself, and so new tables are added. DØ will take advantage of only one of the new tables.

5.1 Structure

Instead of having the low and high luminosity block numbers associated with a file in the `DATA_FILES` table, these values are placed in the `DATAFILE_LUMBLOCK` table as shown in the diagram below.
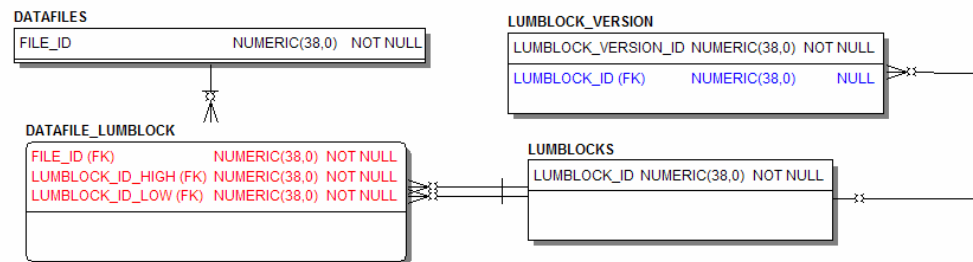

Figure 3: Luminosity block information

The other tables are for storing CDF specific luminosity information and would not be used at DØ. So this diagram is not quite accurate as the foreign keys `LUMBLOCK_ID_HIGH` and `LUMBLOCK_ID_LOW` would be replaced by regular non-null numeric data.

5.1.1 CDF Structure

Note that CDF has an additional table called `LUMBLOCK_VERSION_TYPES`. It has two columns: `LUMBLOCK_VERSION_ID` and `LUMBLOCK_VERSION_DESCRIPTION` (the latter will have values of *accelerator* and *livetimes*). `LUMBLOCK_VERSION_ID` will be a foreign key

into `LUMBLOCK_VERSION` table. Furthermore, `LUMBLOCK_VERSION`'s primary key will be a composite of `LUMBLOCK_ID` and `LUMBLOCK_VERSION_ID`.

5.2 Implementation

The DB server methods and applications that fill in the high and low luminosity block numbers need to now fill the `DATAFILE_LUMBLOCK` table, but only for *raw* data-tier files. Queries for the luminosity block information must also use that table.

Note that the DØ production farm has been filling in high/low luminosity block information for non-raw data-tier files (*e.g.* thumbnails). Presumably, these values are derived from the high/low luminosity block limits from the file's parents, but there's no real guarantee that this was done correctly (the luminosity tools ignore luminosity block information for non-raw data-tier files). As specified above, `DATAFILE_LUMBLOCK` should only be filled for data-tier raw files to avoid confusion.

5.3 Schema Migration

For DØ, the values from the removed `LUM_MIN` and `LUM_MAX` attributes of `DATA_FILES` should be copied into the `LUMBLOCK_LOW_ID` and `LUMBLOCK_HIGH_ID` attributes of the new `DATAFILE_LUMBLOCK` table.

6   DATA_FILES_RAW

In the current SAM DB schema, FILE_LINEAGES holds parent and child information about a file (what file was run over to produce the current file and what files have been produced by running over the current file). This information is essential for luminosity determination and other tasks. To make some luminosity (and perhaps other) tasks easier, a new table is introduced that allows for a direct determination of the raw file parent of a given file without having to traverse intermediate files. The structure of this design is shown below.
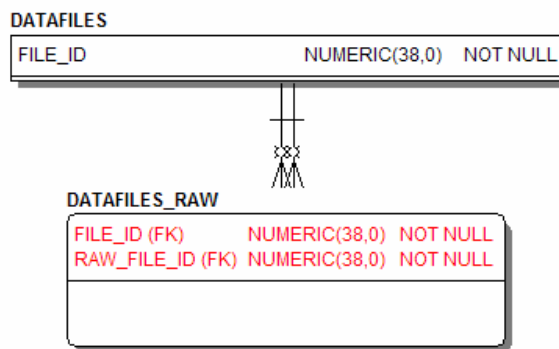
DATAFILES

FILE_ID	NUMERIC(38,0) NOT NULL

DATAFILES_RAW

FILE_ID (FK)	NUMERIC(38,0) NOT NULL
RAW_FILE_ID (FK)	NUMERIC(38,0) NOT NULL

Figure 4: One step to RAW files scheme

6.1 Implementation

DB server methods and applications that insert files into the database will have to traverse the file lineage and fill the `DATA_FILES_RAW` table appropriately. Queries can be added in the future when they are developed for luminosity or other applications.

Obviously files with data-tier of *raw* should not be inserted in this table.